05

Extendiendo una excepción

Transcripción

[00:00] Bueno, ya hemos visto cuál es la diferencia entre Throwable, Error, Exception. Ahora vamos a hacer un pequeño overview de lo que hemos visto hasta ahora, perfecto. Tenemos el stack. En el stack tenemos los métodos, los métodos pueden lanzar excepciones. Yo consigo tratar esas excepciones con try catch. Y consigo lanzar mis propias excepciones con throw.

[00:31] Ahora, yo también puedo crear mis propias excepciones. Pero no puedo extenderlas directamente de Throwable. ¿Por qué? Porque Throwable divide todo en dos grandes grupos que son Exception y Error. Hasta ahí todo bien, y sabemos que hay una jerarquía entre las excepciones que estamos viendo ahora.

[00:51] Y la duda con las que nos quedamos en el video anterior fue por qué yo necesito extender de RuntimeException y no de Exception. Voy a hacer el experimento en el código. Sí, aquí está MiException. Y en lugar de extender de RuntimeException, voy a extender de Exception para ver qué qué es lo que pasa.

[01:18] Lo primero que yo he visto aquí, que ha saltado es que salió una x aquí en flujo.java, eso ya me da un síntoma de que algo no está compilando aquí. Entro aquí, y en efecto, yo tengo aquí ahora que MiException en el catch no está compilando. ¿Y cuál es el problema? Yo voy a borrar esto de aquí, porque yo creo que no está compilando porque método 2 ya no lanza MiException.

[01:47] Entonces voy a descomentar esto para que él lance MiException. Y él no está compilando ni siquiera aquí mi método 2. ¿Por qué? Porque me dice que esta excepción no está siendo atrapada, no está siendo tratada. ¿Pero por qué no? Si en catch, que está aquí yo estoy de hecho, atrapando MiException. ¿Cuál es el problema aquí si ahora estoy extendiendo de Exception?

[02:19] Voy a dar un paso hacia atrás. Voy a poner nuevamente RuntimeException solo para comprobar que mi código ya compilaba antes. Voy a regresar aquí a flujo. Ahora con pila, entonces el problema es cuando extiendo de Exception. Un paso hacia delante nuevamente. Y vamos a averiguar por qué es que no compila.

[02:43] Si en la primera categoría estábamos en los RuntimeException, si ahora estamos en el grupo de los Exceptions, yo necesito ser un poco más explícito en este momento a la hora de tratar con MiException. ¿A qué me refiero yo? Ahora no solamente debo decirle al código que si ejecuta esto, quizás de MiException.

[03:12] Ahora como es un exception propiamente dicho, ya no extiende de RuntimeException sino desde Exception, lo que necesito decirle ahora es que él puede lanzar throws, diferencia con throw, throws, él puede lanzar MiException. Esa es la diferencia. ¿Por qué? Yo aquí necesito explícitamente en la firma del método, decirle: "método 2, lanza MiException". Yo estoy seguro.

[03:48] Estoy totalmente seguro que él puede lanzar mi excepción, a diferencia del try catch, que el try catch es una estructura básicamente si es que él lanza esa excepción, que no sé si lo haga pero si la lanza, atrápala. Por ejemplo si yo ejecuto método 2 aquí fuera, él no va a compilar. ¿Por qué no va a compilar?

[04:11] Porque uno, yo no estoy diciéndole a métopdo 1 que él puede lanzar esa excepción, es la primera cosa. Ahora, si yo le doy un clic aquí, yo tengo dos opciones. La primera es, adiciona la declaración throws, en la firma del método, o sorround con try/catch. ¿Qué significa eso? Rodéalo con un

try/catch, con una estructura try/catch, como es la que yo tenía anterior que es esta de aquí, que sí compilaba sin problemas.

[04:42] ¿Por qué? Porque yo atrapaba esta excepción, la cual yo estoy seguro que este método lanza. Primera cosa. Aquí yo estoy 100% convencido de que él lanza mi excepción o puede lanzar mi excepción si da algún error, pero de alguna forma él puede lanzar esa excepción. No es si por si acaso la lanza. No. Él puede lanzar esta excepción. Es certeza.

[05:07] Entonces, si yo no lo agrego aquí, en el try/catch, y dejo solamente método 2 por ejemplo, yo voy a tener que adicionar aquí, de repente, si yo no quiero usar try/catch, de igual forma, throws MiException y ahí compila este método. ¿Pero cuál es el problema? Sigue subiendo. ¿Por qué?

[05:31] Porque main llama a método 1, entonces main ya necesitaría tratar con MiException. Entonces se dan cuenta cómo la bomba que explota aquí abajo va subiendo, subiendo, subiendo, si es que yo no le doy un tratamiento en el try/catch. Yo podría darle un tratamiento en try/catch aquí. Por ejemplo aquí Eclipse me dice: "Sorround with try/catch". Si le doy clic, automáticamente él me va a generar ese bloque try/catch.

[05:58] Aquí va a hacer un comentario, que este catch block fue generado automáticamente, pero está bien, es lo que yo esperaba, y aquí ya tiene un tratamiento de MiException, entonces básicamente esa es la diferencia si yo extiendo de RuntimeException o extiendo directamente de Exception.

[06:23] En una yo preciso decirle a Java que puede, en una me refiero a RuntimeException, en RuntimeException yo necesito decirle que puede ser, puede que sí, puede que no, pero puede ser que en algún momento lance ese tipo de excepción. En Exception, yo le digo "él va a lanzar esa excepción, cualquier cosa él va a lanzar esa excepción".

[06:47] Entonces es una forma ya digamos de verificar ese comportamiento usando Java. Este tema de las excepciones, en realidad no solamente con

respecto a Java. Lo único que sí es con respecto a Java es esta división entre Exception y Error, ese tratamiento si es todo con respecto a Java, la jerarquía si es puramente Java, pero cómo manejar las excepciones y cómo ya se propagan en la pila de ejecución.

[07:16] Si es de cualquier lenguaje de programación, Python, Ruby, etcétera, todo eso. Entonces ahora ya tenemos buenas razones para elegir o Exception o RuntimeException a la hora de extender, entonces recapitulando un poco aquí, a lo que ya tenemos.

[07:36] Sabemos que las excepciones de Runtime extienden también de Exception y a la vez de Throwable, pero Java nos obliga a declarar en la firma del método o un método try/catch, si nosotros queremos extender directamente de Exception. Esto tiene una explicación y la vamos a ver en el siguiente video.